REMARKS

Claims 1-9 and 11-14 are pending in the application. Claims 1, 5 and 8 have been amended by the foregoing amendment.

Claims 1, 5 and 8 stand rejected under 35 U.S.C. § 112, 1st paragraph as failing to comply with the enabling requirement. It is believed the amendments to these claim overcome this rejection. Applicants request withdrawal of this rejection.

Claims 1-9 and 11-14 stand rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over U.S. Patent No. 6,338,078 ("Chang").

Exemplary embodiments are directed to a method for controlling the order of datagrams being processed by at least one processing engine. As recited in amended claim 1 for example, the at least one processing engine includes at least one input port and at least one output port, wherein each datagram or each group of datagrams has a ticket associated therewith by a ticket dispenser. Processors in the processing engine, once they become available, take the next ticket from the ticket and use it to control the order of the datagram or group of datagrams at the at least one input port of the processing engine and at the at least one output port of the processing engine.

Chang is concerned with preserving packet order in a computer network system. As described (col. 2, lines 41-53), other prior art solutions to queuing inbound packets and distributing them to multiple processors did not take account of scalability nor the preservation of the all-important packet order. Chang makes multiple references to the importance of

preserving packet order (see for example, col. 1, lines 52 and 63; col. 2, lines 43-44, 47 and 49; and col. 6, lines 7-12).

Chang also indicates (col. 4, lines 56-67 and col. 5, lines 27-47) that single CPU/single thread processor systems are effectively locked out of continuing with their current activity when packets are randomly (i.e. unpredictably) received at interrupt level. This is also true for multi-CPU processors operating under single thread operating conditions (col. 5, lines 48-65). When a packet arrives, the CPU responsible for handling the received packet is forced to cease current activity in order to process the arriving packet. In a single processor engine, all other processing ceases for the duration (of processing the arriving packet). In a multiple CPU engines, not only does the responsible CPU cease its current operation but other CPUs operating in concert with that CPU cannot continue their current task even if the other CPUs are not required to process an arriving packet. Therefore, the processing engine as a whole is subjected to an unpredictable impact.

Chang employs multiple (N) CPUs, multiple (N) threads and multiple (N) queues with one thread and one queue being associated for each CPU (as illustrated in Figure 3). Incoming packets are distributed to the respective queues on the basis of their respective MAC (or other) addresses which are contained within the packet header. This distribution is achieved by using a hashing function (50) to allocate packets to the queues (62-68), the packets being delivered to those queues from respective device drivers (42).

Chang states specifically that "techniques exist for ensuring packets being received from a client are sequenced properly . . ." regardless of protocol (col. 6, lines 15+). The method of

Chang simply "applies this sequencing in the context of improved parallelization of multiple CPUs in a network system" (col. 6, lines 18-20). Chang does not introduce a new method or technique *per se* for queuing packets to preserve their order but merely applies *known* techniques to *multithreaded* processor systems, where there is one thread and one queue per processor.

According to one implementation of Chang (col. 6, lines 53 et seq.), the hashing function utilizes MAC addresses contained in the packet headers to hash each of the MAC addresses into the plurality of queues so that packets associated with a given device will all be handled in the same particular queue. In an extreme case, if all packets had the same MAC address, they would all go into one queue and be handled by one CPU while the other CPUs remain idle. This is an inefficient utilization of processing resources (contrary to the objects of exemplary embodiments). Chang is said not to be limited to the use of MAC addresses but extends also to other components of the protocol stack (as illustrated in Figure 4 of Chang), none of which appears relevant to the exemplary embodiments.

Chang fails to describe how the header address is used in the hashing function to preserve packet order. According to Chang (col. 2, lines 55-67), when a queue is started, a thread is scheduled to process packets on that queue. When all of the packets have been processed, the thread becomes dormant. The queue waits for the next packet to arrive thus operating in a passive manner as Chang deals with the whole queue at a time. The processors (CPUs) of Chang are not concerned with allocation of incoming packets to the respective queues. Furthermore, if a queue is associated with a respective thread and a respective CPU and a whole queue is processed at a time, the single CPU handling the queue must process all of the packets in that

queue. There is no possibility for other CPUs to assist in handling and processing of the packets in a particular queue.

In contrast, according to exemplary embodiments, any processing element can process any packet in the queue and can distribute that packet to any client device in accordance with the packet header.

Change operates a "sort on arrival" approach. Exemplary embodiments enable incoming packets to arrive in an unpredictable manner and to be placed in a queue. Each of the plurality of processing elements, upon completion of a current operation, can proactively take the next packet in the queue as identified by the ticket associated with it (packet) by the ticket dispenser. Change fails to disclose a ticket dispenser adapted to associate a ticket with each incoming datagram.

In exemplary embodiments, incoming packets are allocated tickets by a ticket dispenser where the ticket represents the arrival time of the packet. The packets wait in queue while processing elements complete processing preceding packets. Upon completing the processing of a preceding packet, the processing element processes the packet associated with the next ticket. There is no need for the processing element to process a whole batch or queue of tickets as in Chang before seeking the next ticket. All the processing elements operate in this manner. Therefore, although the packets are fetched from the queue in their arrival order, they may be processed faster (or slower) than others, depending the nature of the process. As a result, packets may actually exit the processor in an order that is different from the arriving order/time.

Exemplary embodiments, however, use the ticket to control the order of the packets at the at least one input port and the at least one output port.

Chang fails to disclose or suggest the processing elements, once they become available, taking the next ticket to process the associated datagram (packet). While Chang mentions sequencing (col. 6, lines 1-32), there is no description of a ticket dispenser or tickets and no mechanism for sequencing is described.

Other portions of Chang (col. 6, lines 33-50 and col. 7, lines 1-5) describe packet distribution being performed based on MAC (or other) address data in the packet header as determined by the hashing unit 50 (Fig. 3). There is no reference to a packet being allocated to a particular queue based on ticket allocation by a ticket dispenser. Chang (col. 7, lines 1-5) merely refers to other address options instead of MAC address but the use of other address types does not require tickets and a ticket dispenser.

Chang processes all the packets in a queue and then the CPU responsible for processing the packets in its queue has no more packets to process until more packets arrive and are sent/placed in a queue for this CPU by a hashing function. Chang does not permit a CPU to begin processing packets in other queues (i.e. other than the queue assigned/associated to this CPU).

At least for these reasons, it is believed that claims 1, 5 and 8 are allowable over the teachings of Chang. The remaining claims (i.e. claims 2-4, 6, 7, 9 and 11-14), all of which depend on one of allowable independent claims 1, 5 and 8 are also allowable.

All of the rejections having been overcome, it is believed that this application is in condition for allowance and a notice to that effect is earnestly solicited. Should the Examiner have any questions with respect to expediting the prosecution of this application, she is urged to contact the undersigned at the number listed below.

Respectfully submitted,

Potomac Patent Group, PLLC

By: _____

Krishna Kalidindi
Reg. No. 41,461

Potomac Patent Group PLLC
P.O. Box 270
Fredericksburg, VA 22404
703-893-8500

Date: March 7, 2007